Πανεπιστήμιο Ιωαννίνων Τμήμα Πληροφορικής και Τηλεπικοινωνίων
Αλγόριθμοι και πολυπλοκότητα

Το πρόβλημα της μέγιστης υποακολουθίας

Κώστας Άγγελος

2 Νοεμβρίου 2022

Περίληψη

Με τον όρο πολυπλοκότητα ορίζουμε την διαδικασία υπολογισμού ενός τύπου ο οποίος προσδιορίζει τον συνολικό χρόνο που απαιτείται για την εκτέλεση ενός συγκεκριμένου αλγορίθμου για είσοδο δεδομένων μεγέθους (n). Στην συγκεκριμένη εργασία αναλύονται τρείς αλγόριθμοι που λύνουν το πρόβλημα της μέγιστης υποακολουθίας. Για κάθε έναν ξεχωριστά υπολογίζουμε την πολυπλοκότητα, τονίζοντας έτσι την διαφορά σε ταχύτητα λόγο της επιλογής ενός ικανότερου αλγόριθμου.

1. Εισαγωγή

Το πρόβλημα ουσιαστικά είναι το εξής : Δεδομένου μια ακολουθία ακεραίων τιμών μας ζητείται να βρούμε την υποακολουθία με το μεγαλύτερο άθροισμα. Στην συγκεκριμένη αναφορά παρουσιάζονται τρείς αλγόριθμοι που λύνουν το πρόβλημα :

1) Brute Force Algorithm

Αποτελεί τον πιο απλό τρόπο υπολογισμού της υποακολουθίας γι' αυτό και έχει πολυπλοκότητα ίση με $O(n^3)$.

2) Prefix Algorithm

Αποτελεί μια καλύτερη προσέγγιση με πολυπλοκότητα ίση με $O(n^2)$.

3) Kadane Algorithm

Αποτελεί τον πιο γρήγορο τρόπο επίλυσης του προβλήματος με πολυπλοκότητα ίση με *O(n)*.

2. Πειραματικά αποτελέσματα

Για την καταγραφή των πειραματικών αποτελεσμάτων χρησιμοποιήθηκε το εξής setup :

- IDE : PyCharm 2022.2.3
- Γλώσσα προγραμματισμού : Python 3.10.8
- Βιβλιοθήκη γραφικών : matplotlib

Χαρακτηριστικά υπολογιστή :

- CPU : AMD Ryzen 5 3400g
- RAM : 32 GB DDR4 (2933 MHz)

Παράμετροι πειραμάτων :

Για την μέτρηση του χρόνου εκτέλεσης των αλγορίθμων χρησιμοποιήθηκαν τυχαίοι αριθμοί με εύρος το [-100,100]. Για κάθε έναν αλγόριθμο χρησιμοποιήθηκε πλήθος από διάφορα *N*. Στην συγκεκριμένη αναφορά έχουμε: $N \in \{100,500,1.000,5.000\}$.

Παρακάτω παρουσιάζονται τα αποτελέσματα του χρόνου εκτέλεσης (σε Seconds) για διαφορετικό αριθμό από *N*.

Ένα sample της εξόδου του προγράμματος :

```
Brute Force Approach --> (408, 0, 7) for time: 0.009563

Brute Force Approach --> (1260, 210, 407) for time: 1.105869

Brute Force Approach --> (1878, 203, 982) for time: 9.449848

Brute Force Approach --> (2326, 2838, 3352) for time: 1333.939364

Prefix Approach --> (408, 0, 7) for time: 0.000979

Prefix Approach --> (1260, 210, 407) for time: 0.017453

Prefix Approach --> (1878, 203, 982) for time: 0.073584

Prefix Approach --> (2326, 2838, 3352) for time: 1.826392

Kadane Algorithm Approach --> (408, 0, 7) for time: 4.7e-05

Kadane Algorithm Approach --> (1260, 210, 407) for time: 0.000178

Kadane Algorithm Approach --> (1878, 203, 982) for time: 0.000381

Kadane Algorithm Approach --> (2326, 2838, 3352) for time: 0.001597
```

Εικόνα 1. Output κυρίως προγράμματος

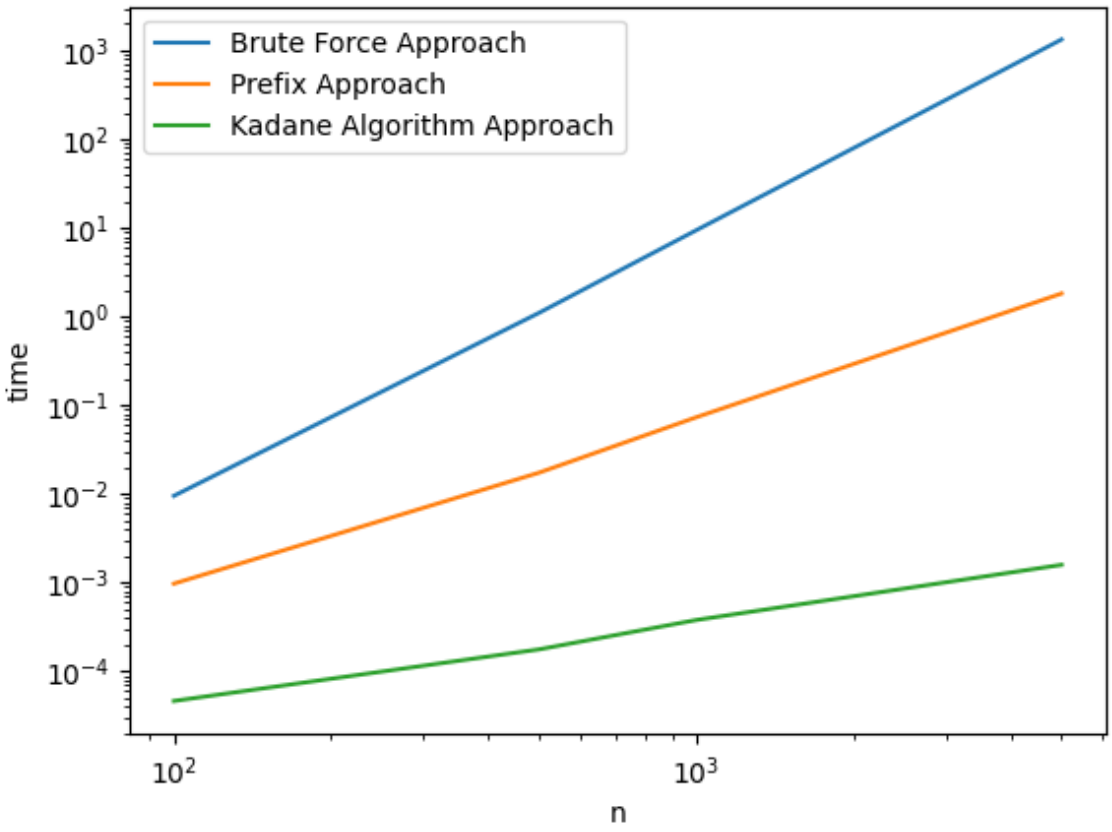
```
PS C:\Users\Leader\Desktop\alco_assignment_1> py tests.py
...
----------------------------------------------------------------------
Ran 3 tests in 0.000s

OK
```

Εικόνα 2. Output των tests

	N=100	N=500	N=1.000	N=5.000
Brute Force Algorithm	0.009563	1.105869	9.449848	1333.939364
Prefix Algorithm	0.000979	0.017453	0.073584	1.826392
Kadane Algorithm	4.7e-05	0.000178	0.000381	0.001597

Πίνακας 1. Πίνακας αποτελεσμάτων



Εικόνα 3. Οπτικοποίηση αποτελεσμάτων (matplotlib)

2. Συμπεράσματα

Από το συγκεκριμένο πρόβλημα αλλά και τις μετρήσεις που εκτελέσαμε φαίνεται η χρησιμότητα σε χρόνο και χώρο από την επιλογή ενός ικανού και γρηγορότερου αλγόριθμου σε σχέση με απλούστερες υλοποιήσεις. Συγκεκριμένα υπάρχει τεράστια διαφορά στο performance ιδίως εάν χρησιμοποιήσουμε μεγάλο αριθμό σε N. Για παράδειγμα για N=10.000 ο brute force μπορεί να διαρκέσει έως και 3.5 ώρες μέχρι να βρει λύση για το πρόβλημα (ανάλογα πάντα και με τα χαρακτηριστικά στα οποία διεξάγεται το πείραμα). Για την συγκεκριμένη αναφορά το εύρος του N επιλέχθηκε με γνώμονα να υπάρχει ισορροπία μεταξύ ταχύτητας και efficiency.